2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

# Adaptive Fireworks Algorithm

Junzhi Li, Shaoqiu Zheng and Ying Tan

*Abstract*— **In this paper, firstly, the amplitude used in the Enhanced Fireworks Algorithm (EFWA) is analyzed and its lack of adaptability is revealed, and then the adaptive amplitude method is proposed where amplitude is calculated according to the already evaluated fitness of the individuals adaptively. Finally, the Adaptive Fireworks Algorithm (AFWA) is proposed, replacing the amplitude operator in EFWA with the new adaptive amplitude. Some theoretical analyses are made to prove the adaptive explosion amplitude a promising method. Experiments on CEC13's 28 benchmark functions are also conducted in order to illustrate the performance and it turns out that the AFWA where adaptive amplitude is adopted outperforms significantly the EFWA and meanwhile the time consumed is not longer. Moreover, according to experimental results, AFWA performs better than the Standard Particle Swarm Optimization (SPSO).**

## I. Introduction

### A. Fireworks Algorithm

THE Fireworks Algorithm (FWA) [1] is a newly developed evolutionary algorithm. Like other evolution algorithms, it also aims to find the vector with the best (usually minimum) fitness in the search space. Inspired by real fireworks, the main idea of the FWA is to use the explosion of the fireworks to search the feasible space of the optimization function, which is a brand new search manner. Its initialization step is simple: choose some fireworks randomly in the search space and evaluate their fitness. Similar to other evolution algorithms, its iteration contains two steps: breeding new individuals and selection. In breeding, for each of the firework, it generates a certain number of explosion sparks within a certain amplitude in an explosion way and a certain number of Gaussian sparks along the direction towards the origin. The explosion sparks are generated randomly within the hypersphere with the firework as its center and the amplitude as its radius. The explosion amplitude and the number of explosion sparks of a firework is calculated according to the fitness of it and other fireworks. Then, all the sparks generated and the fireworks are selected by their fitness and diversity. By iteration, the FWA gets individuals with better and better fitness.

The Enhanced Fireworks Algorithm (EFWA) [2] is an improved version of the FWA. In the EFWA, many operators in the conventional FWA are improved or corrected, such as the way explosion amplitude is calculated, the way new explosion or Gaussian sparks are generated and the way of selection, etc. Hence, our improvement in this paper will be applied to the EFWA rather than the FWA.

Junzhi Li, Shaoqiu Zheng and Ying Tan (corresponding author) are with the Key Laboratory of Machine Perception (Ministry of Education), Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China(email: {ljz, zhengshaoqiu, ytan}@pku.edu.cn).

This work was supported by National Natural Science Foundation of China (NSFC), Grant No. 61375119, No.61170057 and No. 60875080.

### B. Other Related Works

As a newly proposed evolutionary algorithm, FWA was tested on 35 functions with other 11 optimization algorithms by Bureerat [3], and the performance of FWA was ranked 6th, which is higher than Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Some researches were conducted to improve the performance of FWA [4] [5]. FWA was also applied to multi-objective problems by Zheng *et.al.* [6]. K. Ding, S.Q. Zheng and Y. Tan [7] proposed a Parallel Fireworks Algorithm based on GPU. Y. Zheng, X. Xu, and H. Ling [8] proposed a hybrid algorithm combining FWA and DE. Gao, Hongyuan, and Ming Diao [9] proposed a Cultural Firework Algorithm (CFA), which combines the idea of Culture Algorithms and FWA. In the practical problem of Nonnegative Matrix Factorization [10] [11] [12], FWA turned out to be efficient especially when the number of dimension is large. Other applications of FWA were introduced in [13] and [14]. Judging from its performance and application potential, the FWA is a new promising algorithm.

### C. Contribution

Both algorithms of FWA and EFWA remain to be improved in many aspects. For example, neither way of calculating the explosion amplitude in the two algorithms are reasonable. Considering the explosion search in FWA and EFWA, the amplitude is a very important factor influencing the performance. In this paper, we first analyze their disadvantages and drawbacks, and reveal that their explosion amplitudes are not adjusted properly.

To improve the mechanism of calculating the amplitude of explosion in the FWA and the EFWA, an adaptive explosion amplitude is proposed. We use the distance between the best firework and a certain selected individual as the explosion amplitude. We analyze the property of our adaptive amplitude and proved that it can adjust itself adaptively according to the search results. By applying adaptive explosion amplitude to the EFWA, a new algorithm called the Adaptive Fireworks Algorithm (AFWA) is proposed. To illustrate the performance of the AFWA and confirm its advantage over EFWA, some experiments are conducted on the CEC13's 28 benchmark functions. From the results we draw the conclusion that the adaptive explosion amplitude is effective in enhancing the local search capability of the fireworks. The AFWA not only beats EFWA, but also outperforms SPSO2007 and SPSO2011 significantly.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

### D. Synopsis

A brief introduction of the EFWA is presented in section II. A review and analysis of the way explosion amplitudes are calculated in the FWA and the EFWA is presented in Section III. In Section IV, we propose the new adaptive amplitude operator. In Section V, the Adaptive Fireworks Algorithm is proposed. In Section VI, experimental environment and results are illustrated. The analysis of the results and some brief discussion are shown in Section VII. The conclusion is drew in Section VIII.

## II. FRAMEWORK OF THE EFWA

Same as the FWA and other evolutionary algorithms, the structure of EFWA can be briefly divided into four main parts:

1) Initialization

The algorithm chooses $m$ points randomly in the search space as the fireworks of the first generation.

2) Explosion

Each firework generates a certain number of sparks within a certain amplitude. The sparks are generated randomly within the hypersphere with the firework as its center and the amplitude as its radius. Same as the FWA, the number and amplitude are calculated according to their fitness. The better a firework's fitness is, the more sparks it generates and the shorter its amplitude is, vice versa. If the location of a spark generated is out of bound, it will be mapped into the search space randomly, which is different from the FWA.

3) Mutation

A kind of so-called Gaussian mutation happens here. Each firework stretches along the direction between the current location of the firework and the location of the best firework in the EFWA instead of along the direction toward the origin in the FWA.

4) Selection

The best individual is always kept as the fireworks of next generation. Other $m-1$ fireworks are randomly chosen among the rest individuals in the EFWA.

The part 2), 3) and 4) are repeated until the termination criteria is met, such as the evaluation times limit or a given precision demand.

Judging from its structure, the EFWA is a typical evolutionary algorithm. We will briefly introduce some operators in the EFWA that we will use later. For convenience, we unified all the notations.

Same as in the FWA, the numbers of explosion sparks of fireworks are calculated as follows:

$$N_i = \hat{N} \cdot \frac{f(X^\sharp) - f(X_i)}{\sum_{j=1}^{m} (f(X^\sharp) - f(X_j))} \tag{1}$$

where $\hat{N}$ is a parameter controlling the overall spark numbers, $X_i$ is the $i$th firework, $X^\sharp$ stands for the firework with the worst (maximum) fitness. In order to avoid the overwhelming effects of fireworks at good locations, the number of sparks is bounded by

$$N_i = \begin{cases} N_{\min} & if\ N_i < N_{\min} \\ N_{\max} & if\ N_i > N_{\max} \\ N_i & otherwise \end{cases} \tag{2}$$

where $N_{\min}$ and $N_{\max}$ are the lower bound and upper bound for the spark numbers.

Same as in the FWA, the explosion amplitudes of the fireworks are calculated as follows:

$$A_i = \hat{A} \cdot \frac{f(X_i) - f(X^*)}{\sum_{j=1}^{m} (f(X_j) - f(X^*))} \tag{3}$$

where $\hat{A}$ is a parameter controlling the overall explosion amplitude, $X^*$ stands for the firework with the best (minimum) fitness.

In addition, a minimal amplitude check is proposed in the EFWA, reads

$$A_i = \begin{cases} A_{\min} & if\ A_i < A_{\min} \\ A_i & otherwise \end{cases} \tag{4}$$

where

$$A_{\min}(t) = A_{\text{init}} - \frac{A_{init} - A_{final}}{evaltimes} \sqrt{(2 * evaltimes - t)t} \tag{5}$$

where $A_{init}$ and $A_{final}$ are two parameters controlling the initial and final minimum explosion amplitude, $t$ is the current evaluation times, $evaltimes$ stands for the total evaluation times allowed.

Algorithm 1 shows how the sparks are generated in the EFWA.

**Algorithm 1** Generating sparks for $X_i$

```
for j = 1 to N_i do
    for each dimension k = 1, 2, ...d do
        sample r from U(0, 1)
        if r < 0.5 then
            sample r from U(−1, 1)
            s_ij^(k) ← X_i^(k) + r · A_i
            if s_ij^(k) < LB or s_ij^(k) > UB then
                sample r from U(0, 1)
                s_ij^(k) ← LB + r · (UB − LB)
            end if
        end if
    end for
end for
return all the s_ij
```

where $N_i$ is the number of sparks generated by $X_i$, $A_i$ is the amplitude of $X_i$, $UB$ and $LB$ stand for the upper bound and lower bound of the search space respectively, $U(a, b)$ stands for the uniform distribution on $[a, b]$.

In step 3 and 4, the EFWA randomly chooses about half of the dimensions in explosion in order to concentrate its search,

since the number of sparks are very limited compared to the dimension number.

Algorithm 2 shows how Gaussian mutation is conducted in the EFWA.

**Algorithm 2** Generating the Gaussian sparks

**for** $j \leftarrow 1$ **to** $NG$ **do**
  Randomly select $i$ from $1, 2..m$
  sample $r$ from $N(0, 1)$
  **for** each dimension $k = 1, 2, ...d$ **do**
    ${G_j}^{(k)} \leftarrow {X_i}^{(k)} + r \cdot ({X^*}^{(k)} - {X_i}^{(k)})$
    **if** ${G_j}^{(k)} < LB$ **or** ${G_{ij}}^{(k)} > UB$ **then**
      sample $r$ from $U(0, 1)$
      ${G_j}^{(k)} \leftarrow LB + r \cdot (UB - LB)$
    **end if**
  **end for**
**end for**
**return** all the $G_j$

where $NG$ is the number of Gaussian sparks, $m$ is the number of fireworks, $X^*$ stands for the best firework, $UB$ and $LB$ stand for the upper bound and lower bound of the search space respectively, $U(a, b)$ stands for the uniform distribution on $[a, b]$, $N(\mu, \sigma)$ stands for the normal distribution with mean $\mu$ and standard deviation $\sigma$.

In summary, we have the complete framework of the Enhanced Fireworks Algorithm, shown in Algorithm 3.

**Algorithm 3** Enhanced Fireworks Algorithm

randomly select $m$ fireworks in the potential space
evaluate their fitness
**repeat**
  calculate $N_i$ according to Eq.(1) and Eq.(2)
  calculate $A_i$ according to Eq.(3), Eq.(4) and Eq.(5)
  for each firework, generate $N_i$ sparks within amplitude $A_i$ according to Algorithm 1
  generate $NG$ Gaussian sparks according to Algorithm 2
  evaluate all the sparks' fitness
  keep the best individual as a firework
  randomly choose other $m-1$ fireworks among the rest individuals
**until** termination criteria is met
**return** the best individual and its fitness

The EFWA significantly improved the performance of FWA, especially on functions whose optimums are far from origin. More importantly, some operators unreasonable in the FWA are corrected in the EFWA. This is the main reason why we apply our improvement to the EFWA rather than the FWA.

## III. Analysis on the Amplitude in FWA and EFWA

Considering the explosion search manner in the FWA and the EFWA, the amplitude of each fireworks is a fatal variable influencing the performance of the algorithm. As shown in Eq.(3), the amplitudes of other fireworks (except for the best one) are calculated according to the difference between their fitness and the best one's. However, the amplitude of the best firework is always 0 in Eq.(3). In the FWA, there is no other operators to deal with this problem, which means the best firework will not contribute to the algorithm, despite its most numerous sparks. Note that according to Eq.(1), the better a firework's fitness is, the more sparks it generates. In the FWA, The best fireworks takes the most and provides the least.

Thus, in the EFWA, in order to make sure the best firework works, a minimal amplitude check is adopted, preventing the amplitude of the best firework from being 0. As is shown in Eq.(4) and Eq.(5), the threshold of the amplitude is a nonlinear decrease function of the generation number. In the EFWA, the threshold is actually the amplitude of the best firework.

However, any preset amplitude, linear decrease or nonlinear decrease, whatever the parameters are set, cannot fit the evaluation function well: in some functions, it decreases too fast, and in others too slow. Decreasing too fast causes the search range converges early before the minima is reached. Decreasing too slow causes the search range is still too large to search precisely even when the minima is already within the search range. In either case, the algorithm performs bad.

The amplitudes of other fireworks in FWA and EFWA are calculated according to Eq.(3) adaptively, while the amplitude of the best firework still remains a big problem. Although it only influences local search of the algorithm, it is a key to the performance. The amplitude of the best firework need to be adjusted automatically in order to fit all evaluation functions.

## IV. Adaptive Explosion Amplitude

In this section, we propose an adaptive method, using already generated sparks to calculate the explosion amplitude of the best firework. We use the information obtained in this generation to calculated the amplitude of the best firework in the next generation. Considering the selection in EFWA, the best firework in next generation is the best individual found (could be a spark generated or a firework) in this generation. As we already know, all the amplitudes of other fireworks (in the next generation) are calculated according to the difference between their fitness and the best firework. The main problem is to get a reasonable amplitude of the best firework. For convenience, we only consider one firework within this section.

To calculate an adaptive amplitude, we choose an individual and use its distance to the best individual, which is the firework in next generation, as the amplitude of the next explosion. The individual we choose subjects to the following conditions:

1) Its fitness is worse than the firework of this generation.

2) Its distance to the best individual (the firework of next generation) is minimal among all individuals subjecting to 1).

Namely,

$$\hat{s} = \arg\min_{s_i}(d(s_i, s^*)) \tag{6}$$

with constraint

$$f(s_i) > f(X) \tag{7}$$

where $s_i$ stands for all sparks generated by the firework, $s^*$ stands for the best individual among all sparks and the firework, $X$ stands for the firework , $d$ is a certain measurement of distance. Note that the algorithm always choose the best individual as the firework in next generation.

Condition 1) requires that the difference (in fitness) between this individual and the best individual is bigger than that between the firework and the best individual, reads

$$f(\hat{s}) - f(s^*) > f(X) - f(s^*) \tag{8}$$

An intuitive understanding of this inequality is it assures that the scale on the domain of the evaluation function within the range $d(\hat{s}, s^*)$ is at least bigger than the improvement made in this generation. We aim to find a better location $\tilde{s}$ in the next explosion such that

$$f(s^*) - f(\tilde{s}) > f(X) - f(s^*) \tag{9}$$

by estimating the potential range $d(\tilde{s}, s^*)$ with $d(\hat{s}, s^*)$. The algorithm makes a correspondence between the range and the domain. Some deeper consideration and its properties will be discussed after we present the complete algorithm.

On the other hand, condition 2) helps to make sure the amplitude converges. If a farther individual subjecting to condition 1) is chosen, the amplitude could be, in the worst case, double the amplitude of this explosion. Under that circumstances, there is no guarantee the amplitude won't be locked on the maximum value (the range of the search space, for example). While, on the contrary, choosing the nearest individual is quite safe because if the function is regular locally and the sparks are numerous enough, the minimum distance would be at most slightly longer than the amplitude in this generation. Although we cannot promise the amplitude decreases every time, but with big iteration times and numerous sparks, it converges in general, as is shown in Fig.1.

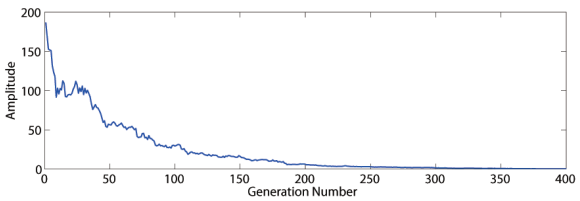


Fig. 1. Adaptive Amplitude on Sphere Function

For example, in Fig. 2, the red dot with fitness 1.0 is the firework of this generation, and yellow dots are the sparks generated by it. Obviously, we will choose the yellow dot with fitness 0.7 as the firework in the next explosion. According to Eq.(6) and Eq.(7), we choose the individual whose fitness is 1.1 as $\hat{s}$ and use its distance to the 0.7 individual ($s^*$) as the amplitude of next explosion. Without condition 1) we will choose the 0.8 individual, which makes the algorithm converges too fast. Without condition 1) we will choose the 1.5 individual, which makes the algorithm doesn't converge at all. If the evaluation function changes regularly, with numerous sparks, there is good chance we find the individual with fitness 0.3 in the next explosion, if 0.7 is not close to local minima.

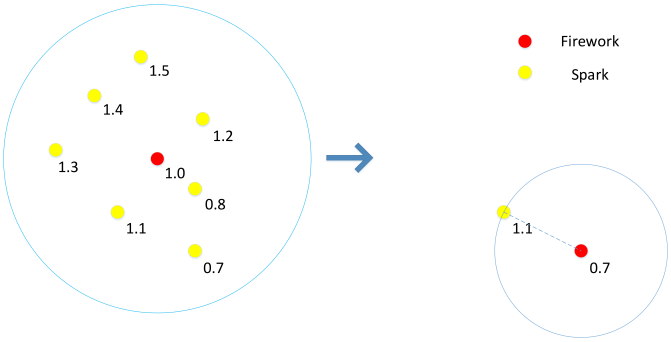


Fig. 2. An Example of How the Adaptive Amplitude is Calculated

It's clear that this algorithm does not care either the scale of the range or the domain. Rather, it detects the relationship between the scale of the range and the scale of the domain adaptively without any more evaluation. Changing the scale of either will not influence the performance. In a sense, it provides the gradient information of the evaluation function.

Considering the way fireworks explode in the Enhanced Fireworks Algorithm, where they explode in each dimension independently, we use infinity norm as the distance measure, namely the maximum difference among all dimensions. Besides, in order to further slow down the convergence rate and improve the global search, the adaptive amplitude calculated above is multiplied by a certain coefficient (usually bigger than 1). Finally, considering the sparks of each explosion is limited, in order to minimize the influence of very bad luck (for example, every spark is worse than the firework, in which case the amplitude shrinks very fast, or on the contrary the amplitude doubled last time), we also adopt a simple smoothing mechanism, which uses the average of the amplitude calculated above and the amplitude of this generation as the amplitude.

The complete algorithm of calculating the amplitude is shown in Algorithm 4.

In Algorithm 4, $UB$ and $LB$ stand for the upper bound and lower bound of the search space respectively, $s_1 ... s_n$ stands for all sparks generated by the firework in generation $g$, $X$ stands for the firework in generation $g$, $s^*$ stands for the best individual in generation $g$, namely the firework in generation $g + 1$.

The parameter $\lambda$ has a great impact on the performance of the algorithm. In a sense, it controls the balance between global search and local search. If $\lambda$ is too small, the adaptive

**Algorithm 4** Calculate the Adaptive Amplitude for the Firework of Generation $g+1$

$A(g+1) \leftarrow UB - LB$
**for** $i = 1$ **to** $n$ **do**
  **if** $||s_i - s^*||_\infty > A(g+1)$ **and** $f(s_i) > f(X)$ **then**
    $A(g+1) \leftarrow ||s_i - s^*||_\infty$
  **end if**
**end for**
$A(g+1) \leftarrow \lambda \cdot A(g+1)$
$A(g+1) \leftarrow 0.5 \cdot (A(g) + A(g+1))$
**return** $A(g+1)$

amplitude converges too fast to a local minima without searching the neighbourhood. While if it is too big, the adaptive amplitude doesn't converge. Generally speaking, as long as it converges stably, the bigger the better. In experiment we usually use $\lambda = 1.3$ empirically.

The computational complexity of Algorithm 4 is $O(n)$, which means it doesn't add much cost. Actually, compared to other operators such as generating sparks, $O(n)$ is not dominant.

For most optimization problems, the process of searching by only one firework can be briefly divided into 3 stages (note that they are not strictly distinguished):

1) Global search. At the beginning, the algorithm doesn't have any information about the evaluation function, so it has to explore globally to decide which region is comparatively promising for further exploitation. In our algorithm, the amplitude of the firework is set to the range of the search space at the beginning, and the sparks it generates will be distributed in the whole search space. In this way, the algorithm detects the rough information about which region is good and which is not. Then the algorithm will choose the best individual among all the sparks and the firework. As far as the amplitude is concerned, it's really hard to predict whether it will become longer or shorter. There are two possible cases, roughly speaking, as shown in Fig. 3.

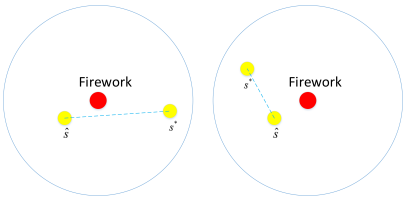


Fig. 3. Two Cases in Explosion. Left: $\hat{s}$ and $s^*$ are on the different sides of the firework; Right: $\hat{s}$ and $s^*$ are on the same side of the firework.

In the first case, $\hat{s}$ and $s^*$ are on the different sides of the firework. To the limited information we have, we can assume in reason that the firework is in the same local region as $\hat{s}$ and $s^*$, and that the $d(\hat{s}, s^*)$ gives the longest estimate of the region's scale. In this case, the amplitude will most probably become longer and the $s^*$ will walk a longest step toward the potential local best.

In the second case, $\hat{s}$ and $s^*$ are on the same side of the firework. Recall that the fitness of $\hat{s}$ we use is worse than the firework, while $s^*$ is better. To the limited information we have, we can assume in reason that the firework is not in the same local region as $\hat{s}$ and $s^*$, and that the region containing $s^*$ is more promising than that containing the firework ,and that the $d(\hat{s}, s^*)$ gives the longest estimate of the region's scale. In this case, the amplitude will most probably become shorter, in order to give up the region of the firework and search more efficiently.

In summary, $d(\hat{s}, s^*)$ always gives the longest possible estimate of the scale of the local region containing $s^*$ and keeps the search most efficient.

2) Local search. When the firework enters a certain local region, which we can assume as a bowl region, the main task is to walk toward the bottom as fast as possible. When the amplitude is still much shorter than the distance between the firework and the bottom, we can assume that the neighbourhood of the firework is monotonous, which means $\hat{s}$ and $s^*$ will undoubtedly on the different sides of the firework. From the monotonic we can also assume that the $s^*$ is very close to the border of the firework's amplitude, because there is no farther spark in its direction. Then, it is most likely that $d(\hat{s}, s^*) > d(X, s^*)$, as shown in Fig. 4.

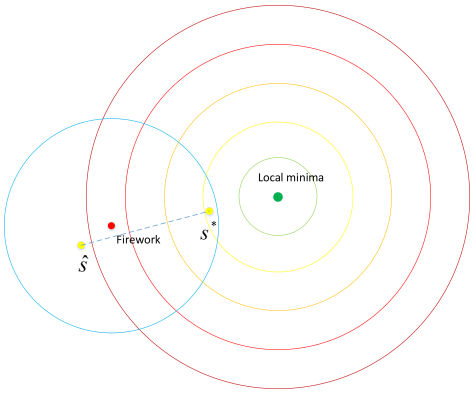


Fig. 4. Local Search

As shown in Fig. 5, the experimental result also supports this conclusion.

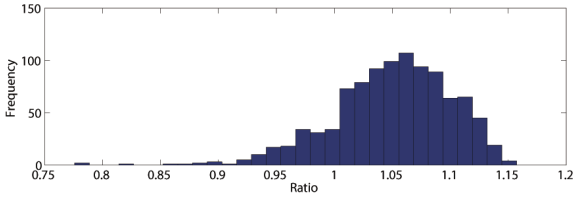


Fig. 5. A Histogram of the Ratio by Which the Amplitude Increases at Stage 2)

In summary, the amplitude will become longer and longer in stage 2) to fasten the search as long as the firework is still far from the local minima.

3) Refine search. At the end of the search, when the local minima is already within the amplitude of the firework, the

algorithm need to search more precisely than in the above stages. In this case, the $s^*$ is not the farthest spark, rather it is the spark closest to the local minima. So, contrary to stage 2), the amplitude will most probably become shorter and shorter (shown in Fig. 6) and enable the algorithm to search more and more precisely, unless the local minima is not within the amplitude any longer, which brings the algorithm back to stage 2). Finally, as we have shown in Fig. 1, the amplitude converges after all, and the local minima is certainly reached.

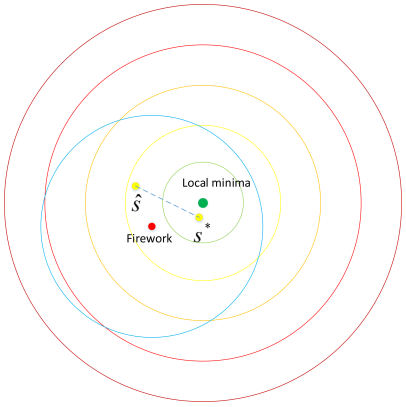


Fig. 6. Refine Search

The properties of our algorithm in all the 3 stages proves it a promising global-to-local search algorithm.

There are some extreme cases for the algorithm:

1) It is possible that all the sparks' fitness is worse than the firework's. It is most likely to happen in stage 3). Usually it implies that the firework is quite close to the local minima, while the amplitude is too large. In our algorithm, if it happens, $\hat{s}$ will just be the closest spark to the firework, and the amplitude in next generation will become shorter than this generation. When the amplitude is reduced to a reasonable level (there is certainly a better location in a very small neighbourhood unless the local minima is already reached), the algorithm will go on to work normally.

2) It is possible that all the sparks' fitness is better than the firework's. It is most likely to happen in stage 1). According to Algorithm 4, the amplitude will be set to the range of the search space. It is still reasonable because the firework can be considered a local maxima in this case, which means search around it will be meaningless.

So, even in extreme cases, the algorithm has a strong capability of error correction.

The explosion amplitude could be considered as the step size in the Fireworks Algorithm. To the best of our knowledge, the adaptive explosion amplitude we proposed is a brand new method to control the step size in an evolutionary algorithm.

## V. Adaptive Fireworks Algorithm

In this section we will apply the adaptive amplitude to the Enhanced Fireworks Algorithm.

At the beginning after initialization, we set the amplitude of the best firework to the range of the search space. Then, for each generation, we use the method introduced in section IV to calculate the amplitude of the best firework in the next generation.

When there are more than two fireworks adopted in the algorithm, the $\hat{s}$ we look for in section IV could be a spark as well as a firework.

The amplitudes of other fireworks are still calculated according to Eq.(3), and the minimal amplitude check is dropped in our algorithm since it is designed to control the amplitude of the best firework. Except for the amplitude of the best firework, we basically follow the operators in the EFWA.

Algorithm 5 shows the complete version of the Adaptive Fireworks Algorithm.

**Algorithm 5** Adaptive Fireworks Algorithm

```
randomly select m fireworks in the potential space
evaluate their fitness
A* ← UB − LB
repeat
    calculate N_i according to Eq.(1) and Eq.(2)
    calculate A_i (except for A*) according to Eq.(3)
    generate N_i sparks according to Algorithm 1
    generate Gaussian sparks according to Algorithm 2
    evaluate all sparks' fitness
    calculate A* according to Algorithm 4
    keep the best individual as a firework
    randomly choose other m−1 fireworks among the rest individuals
until termination criteria is met
return the best individual and its fitness
```

where $m$ is the number of fireworks, $UB$ and $LB$ stand for the upper bound and lower bound of the search space respectively, $A_i$ stands for the amplitude of each firework, $N_i$ stands for the number of sparks of each firework, $A^*$ stands for the amplitude of the best firework.

We can see from Algorithm 5 that the number of parameters adopted in the AFWA is less than that in the EFWA, since the EFWA adopts 2 parameters in minimal amplitude check, which is now replaced by adaptive amplitude using only 1.

## VI. Experiment

In order to illustrate and compare the performance of AFWA and EFWA, experiments on 28 CEC13's benchmark functions [15] were conducted. The introduction of the 28 functions is shown in Table I. In AFWA and EFWA, $m = 5$, $N_{\min} = 2$, $N_{\max} = 100$, $\hat{N} = 200$, $\hat{A} = 100$ and $NG = 5$. In AFWA, $\lambda = 1.3$. Besides, the results of SPSO2007 and SPSO2011, which is the latest version of the Standard PSO, were adopted as a baseline. The parameters of SPSO2007 are the same as [16], and the results of SPSO2011 are obtained directly from [17] and `http://t.cn/8Fqg1rN`. Evaluation times: 300000, Run times: 51, Dimension: 30. Experiment environment: MATLAB2011b; Win 7; Intel Core i7-2600 CPU; 3.7GHZ; 8GB RAM.

TABLE I

CEC13'S 28 BENCHMARK FUNCTIONS

| | No. | Name |
|---|---|---|
| Unimodal Functions | 1 | Sphere Function |
| | 2 | Rotated High Conditioned Elliptic Function |
| | 3 | Rotated Bent Cigar Function |
| | 4 | Rotated Discus Function |
| | 5 | Different Powers Function |
| Basic Multimodal Functions | 6 | Rotated Rosenbrocks Function |
| | 7 | Rotated Schaffers F7 Function |
| | 8 | Rotated Ackleys Function |
| | 9 | Rotated Ackleys Function |
| | 10 | Rotated Griewanks Function |
| | 11 | Rastrigins Function |
| | 12 | Rotated Rastrigins Function |
| | 13 | Non-Continuous Rotated Rastrigins Function |
| | 14 | Schwefel's Function |
| | 15 | Rotated Schwefel's Function |
| | 16 | Rotated Katsuura Function |
| | 17 | Lunacek Bi_Rastrigin Function |
| | 18 | Rotated Lunacek Bi_Rastrigin Function |
| | 19 | Expanded Griewanks plus Rosenbrocks Function |
| | 20 | Expanded Scaffers F6 Function |
| Composition Functions | 21 | Composition Function 1 (Rotated) |
| | 22 | Composition Function 2 (Unrotated) |
| | 23 | Composition Function 3 (Rotated) |
| | 24 | Composition Function 4 (Rotated) |
| | 25 | Composition Function 5 (Rotated) |
| | 26 | Composition Function 6 (Rotated) |
| | 27 | Composition Function 7 (Rotated) |
| | 28 | Composition Function 8 (Rotated) |

The mean error of the 4 algorithms is presented in Table II.

The ranking of the 4 algorithms' mean error on 28 benchmark functions was calculated, shown in Table III.

A set of T-tests were also conducted to illustrate whether the improvement of AFWA over EFWA is significant. The null hypothesis is the results of the AFWA and those of the EFWA come from distributions with equal means. $H = 1$ indicates that the null hypothesis can be rejected at the 5% level. Table IV shows the $H$ and $p$ values.

Fig. 7 shows the time consumed by each algorithm.

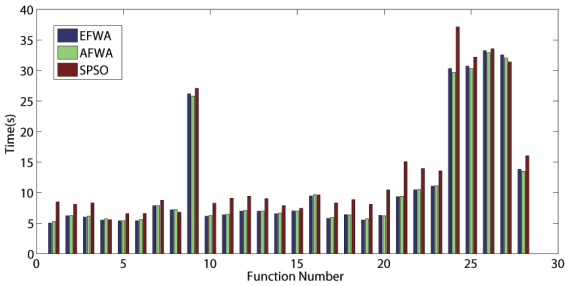


Fig. 7. Time Consumed by Each Algorithm on 28 Functions

## VII. DISCUSSION

According to Table II,III and IV, we can see that AFWA outperformed EFWA significantly: except for function 2 and function 3 where AFWA and EFWA performed almost the same, EFWA only beat AFWA on function 4 and function

TABLE II

MEAN ERROR ON 28 FUNCTIONS

| Fun.\Alg. | SPS02007 | SPSO2011 | EFWA | AFWA |
|---|---|---|---|---|
| 1 | 0.0000E+00 | 0.0000E+00 | 8.4987E-02 | 0.0000E+00 |
| 2 | 6.0817E+06 | 3.3849E+05 | 5.8486E+05 | 7.7759E+05 |
| 3 | 6.6257E+08 | 2.8841E+08 | 1.1588E+08 | 1.4990E+08 |
| 4 | 1.0284E+05 | 3.8643E+04 | 1.2217E+00 | 2.1172E+01 |
| 5 | 0.0000E+00 | 5.4221E-04 | 8.0517E-02 | 5.9500E-04 |
| 6 | 2.5243E+01 | 3.7901E+01 | 3.2152E+01 | 2.8939E+01 |
| 7 | 1.1275E+02 | 8.7916E+01 | 1.4417E+02 | 9.6581E+01 |
| 8 | 2.0952E+01 | 2.0916E+01 | 2.0952E+01 | 2.0910E+01 |
| 9 | 2.9323E+01 | 2.8769E+01 | 2.9843E+01 | 2.5666E+01 |
| 10 | 2.3808E-01 | 3.4019E-01 | 8.4819E-01 | 5.9482E-02 |
| 11 | 6.2592E+01 | 1.0496E+02 | 2.7947E+02 | 1.1153E+02 |
| 12 | 1.1533E+02 | 1.0396E+02 | 4.0648E+02 | 1.7181E+02 |
| 13 | 1.7898E+02 | 1.9386E+02 | 3.5115E+02 | 2.5091E+02 |
| 14 | 1.5877E+03 | 3.9910E+03 | 4.0183E+03 | 2.8938E+03 |
| 15 | 4.3050E+03 | 3.8093E+03 | 4.2843E+03 | 3.9328E+03 |
| 16 | 1.2711E+00 | 1.3066E+00 | 5.7506E-01 | 5.5301E-01 |
| 17 | 9.9841E+01 | 1.1626E+02 | 2.1685E+02 | 1.4416E+02 |
| 18 | 1.7963E+02 | 1.2063E+02 | 1.7235E+02 | 1.7682E+02 |
| 19 | 6.4771E+00 | 9.5101E+00 | 1.2365E+01 | 6.8633E+00 |
| 20 | 1.5000E+01 | 1.3463E+01 | 1.4520E+01 | 1.3100E+01 |
| 21 | 3.3537E+02 | 3.0879E+02 | 3.2833E+02 | 2.9813E+02 |
| 22 | 2.9832E+03 | 4.2988E+03 | 5.1547E+03 | 3.3632E+03 |
| 23 | 6.9666E+03 | 4.8313E+03 | 5.7311E+03 | 4.5682E+03 |
| 24 | 2.8972E+02 | 2.6675E+02 | 3.0495E+02 | 2.7532E+02 |
| 25 | 3.1026E+02 | 2.9928E+02 | 3.3751E+02 | 2.9757E+02 |
| 26 | 2.5725E+02 | 2.8605E+02 | 3.0248E+02 | 2.7057E+02 |
| 27 | 8.1625E+02 | 1.0046E+03 | 1.2165E+03 | 9.8336E+02 |
| 28 | 6.9249E+02 | 4.0132E+02 | 1.2292E+03 | 3.1624E+02 |

TABLE III

RANKING OF MEAN ERROR ON 28 FUNCTIONS

| Fun.\Alg. | SPS02007 | SPSO2011 | EFWA | AFWA |
|---|---|---|---|---|
| 1 | 1 | 1 | 4 | 1 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 4 | 3 | 1 | 2 |
| 4 | 4 | 3 | 1 | 2 |
| 5 | 1 | 2 | 4 | 3 |
| 6 | 1 | 4 | 3 | 2 |
| 7 | 3 | 1 | 4 | 2 |
| 8 | 3 | 2 | 4 | 1 |
| 9 | 3 | 2 | 4 | 1 |
| 10 | 2 | 3 | 4 | 1 |
| 11 | 1 | 2 | 4 | 3 |
| 12 | 2 | 1 | 4 | 3 |
| 13 | 1 | 2 | 4 | 3 |
| 14 | 1 | 3 | 4 | 2 |
| 15 | 4 | 1 | 3 | 2 |
| 16 | 3 | 4 | 2 | 1 |
| 17 | 1 | 2 | 4 | 3 |
| 18 | 4 | 1 | 2 | 3 |
| 19 | 1 | 3 | 4 | 2 |
| 20 | 4 | 2 | 3 | 1 |
| 21 | 4 | 2 | 3 | 1 |
| 22 | 1 | 3 | 4 | 2 |
| 23 | 4 | 2 | 3 | 1 |
| 24 | 3 | 1 | 4 | 2 |
| 25 | 3 | 2 | 4 | 1 |
| 26 | 1 | 3 | 4 | 2 |
| 27 | 1 | 3 | 4 | 2 |
| 28 | 3 | 2 | 4 | 1 |
| Mean | 2.4286 | 2.1786 | 3.3929 | 1.8929 |

TABLE IV
T-TEST ON AFWA VS. EFWA

| Function Number | $H$ | $p$ |
|---|---|---|
| 1 | 1 | 0.0000E+00 |
| 2 | 0 | 9.9028E-01 |
| 3 | 0 | 2.7459E-01 |
| 4 | 1 | 2.3319E-172 |
| 5 | 1 | 0.0000E+00 |
| 6 | 1 | 1.8752E-122 |
| 7 | 1 | 1.1644E-103 |
| 8 | 1 | 0.0000E+00 |
| 9 | 1 | 1.1737E-191 |
| 10 | 1 | 0.0000E+00 |
| 11 | 1 | 2.6480E-40 |
| 12 | 1 | 5.3906E-03 |
| 13 | 1 | 1.0226E-15 |
| 14 | 1 | 8.9976E-18 |
| 15 | 1 | 8.1931E-03 |
| 16 | 1 | 5.7970E-254 |
| 17 | 1 | 8.1730E-72 |
| 18 | 1 | 1.5146E-75 |
| 19 | 1 | 3.0206E-187 |
| 20 | 1 | 1.4371E-259 |
| 21 | 1 | 7.7820E-67 |
| 22 | 1 | 2.6944E-28 |
| 23 | 1 | 8.3888E-22 |
| 24 | 1 | 3.4469E-147 |
| 25 | 1 | 2.2101E-157 |
| 26 | 1 | 7.1659E-88 |
| 27 | 1 | 1.1631E-87 |
| 28 | 1 | 3.0053E-17 |

18. AFWA and EFWA share almost all the parameters in common, so the significant difference in their performance proved that the new adaptive amplitude is significantly effective. More precisely, since the global search in AFWA and EFWA are almost the same, the adaptive amplitude actually improved its local search capability.

According to Table II, AFWA performed much better than SPSO2007 and SPSO2011: AFWA beats SPSO2011 on 18 out of 27 functions and beats SPSO2007 on 16 out of 27 functions (except for function 1 where they are even). According to Table III, judging from the overall performance, AFWA is the best algorithm among all the 4 algorithms.

In terms of computation cost, as is shown in Fig. 3, the time consumed by AFWA and EFWA were very close, which is less than SPSO. The computation cost of calculating the adaptive amplitude is very low($O(n)$), compared to other operators such as generating sparks.

Judging from the performance and the computation cost, the Adaptive Fireworks Algorithm is a very promising, efficient and simple algorithm.

## VIII. CONCLUSION

In this paper, we analyzed the amplitude of explosion in the FWA and the EFWA, then we proposed an adaptive amplitude. The distance of the best firework and a certain individual subjecting to some conditions is employed as the amplitude of the explosion. We analyzed the property of the adaptive amplitude and come to the conclusion that the adaptive amplitude for explosion is a theoretically promising operator. Replacing the amplitude operator in EFWA by adaptive amplitude, we proposed a new algorithm called Adaptive Fireworks Algorithm. According to the experimental results on CEC13's 28 benchmark functions, the performance is greatly improved: the AFWA not only outperforms EFWA but also beats SPSO 2007 and SPSO2011 totally. The results proved that the adaptive amplitude is effective because AFWA greatly improved performance of EFWA, and meanwhile the computation cost of AFWA is not obviously bigger than EFWA.

## REFERENCES


[1] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in Advances in Swarm Intelligence. Springer, 2010, pp. 355–364.

[2] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013, pp. 2069–2077.

[3] S. Bureerat, "Hybrid population-based incremental learning using real codes," in Learning and Intelligent Optimization. Springer, 2011, pp. 379–391.

[4] J. Liu, S. Zheng, and Y. Tan, "The improvement on controlling exploration and exploitation of firework algorithm," in Advances in Swarm Intelligence. Springer, 2013, pp. 11–23.

[5] Y. Pei, S. Zheng, Y. Tan, and H. Takagi, "An empirical study on influence of approximation approaches on enhancing fireworks algorithm," in Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on. IEEE, 2012, pp. 1322–1327.

[6] Y.-J. Zheng, Q. Song, and S.-Y. Chen, "Multiobjective fireworks optimization for variable-rate fertilization in oil crop production," Applied Soft Computing, vol. 13, no. 11, pp. 4253–4263, 2013.

[7] K. Ding, S. Zheng, and Y. Tan, "A gpu-based parallel fireworks algorithm for optimization," in GECCO '13 Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference. ACM, 2013, pp. 9–16.

[8] Y. Zheng, X. Xu, and H. Ling, "A hybrid fireworks optimization method with differential evolution," Neurocomputing, 2012.

[9] H. Gao and M. Diao, "Cultural firework algorithm and its application for digital filters design," International Journal of Modelling, Identification and Control, vol. 14, no. 4, pp. 324–331, 2011.

[10] A. Janecek and Y. Tan, "Using population based algorithms for initializing nonnegative matrix factorization," in Advances in Swarm Intelligence. Springer, 2011, pp. 307–316.

[11] ——, "Iterative improvement of the multiplicative update nmf algorithm using nature-inspired optimization," in Natural Computation (ICNC), 2011 Seventh International Conference on, vol. 3. IEEE, 2011, pp. 1668–1672.

[12] ——, "Swarm intelligence for non-negative matrix factorization," International Journal of Swarm Intelligence Research (IJSIR), vol. 2, no. 4, pp. 12–34, 2011.

[13] W. He, G. Mi, and Y. Tan, "Parameter optimization of local-concentration model for spam detection by using fireworks algorithm," in Advances in Swarm Intelligence. Springer, 2013, pp. 439–450.

[14] S. Zheng and Y. Tan, "A unified distance measure scheme for orientation coding in identification." in Information Science and Technology, 2013 IEEE Congress on. IEEE, 2013, pp. 979–985.

[15] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," 2013.

[16] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in Swarm Intelligence Symposium, 2007. SIS 2007. IEEE. IEEE, 2007, pp. 120–127.

[17] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements," in Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013, pp. 2337–2344.